

6317592

2/5/01

1/15/01

[illegible][illegible]

Copyright Thomas Campana, Jr. 1991

#define ATT\_EMAIL\_FILE  
#define DELIMITER

"TFMOBOX.TMP"  
"End of Telefind Network Message\n"

0443403 120500  
00502T 005340

```
#include <string.h>
#include <time.h>
#include <stdio.h>
#include <dos.h>
#include "saferi.h"
```

```
void main(void)
```

```
{
```

```
    FILE *infile,*outfile;
    char buffer(81),chr,timestr(6),datestr(9);
    char msg_num(4);
    int msg_num_opt = 0;
    char *ptr;
    int x,day,month,line=1,attmail=0;
    time_t t;
```

```
    if ((infile = fopen(ATT_EMAIL_FILE,"rt")) == NULL)
    {
        printf("%s does not exist\n",ATT_EMAIL_FILE);
        exit(0);
    }
```

```
    if ((outfile = fopen("tfmobox.$$$","wt")) == NULL)
    {
        printf("Can't open TFMBOX.$$$\n");
        exit(0);
    }
```

```
    for(;;)
```

```
    {
```

```
        /* get characters from .tmp file */
```

```
        x = 0;
```

```
        do
```

```
        {
```

```
            chr = fgetc(infile);
```

```
            if (feof(infile))
```

```
            {
```

```
                fclose(infile);
```

```
                fclose(outfile);
```

```
                exit(0);
```

```
            }
```

```
            buffer(x++) = chr;
```

```
        }
```

```
        /* until end of line */
```

```
        while (chr != '\n' && x != 80);
```

```
        buffer[x] = '\0'; /* terminate it */
```

```
        if (line == 1)
```

```
        {
```

```
            ptr = strchr(buffer,');');

```

```
            if (ptr-buffer == 2) /* was 3rd character */

```

```
            {
```

```
                sscanf(buffer,"%[~]",msg_num);

```

```
                msg_num_opt = 1;

```

```
                ptr++;

```

```
            }
```

```
            else
```

```
                ptr = buffer;
```

```
            if (*ptr == ':' && *(ptr+1) == '\0')

```

```
                attmail = 1;

```

```
        }
```

```
        if (attmail)
```

```
        {
```

```
            switch(line)
```

```

(
    case 1:
        /*      datestr = mm/dd, timestr = hh:mm      */
        sscanf(datestr,"%d/%d",&month,&day);          */
        /*      get year from pc      */

        t = time(NULL);
        fprintf(outfile,"Date: %s",ctime(&t));
        break;
    case 2:
        fprintf(outfile,"From: %s",buffer);
        break;
    case 3:
        fprintf(outfile,"Subject: %s",buffer);
        fprintf(outfile,"To: <Name here>\n");
        if (msg_num_opt)
            fprintf(outfile,"Message #%s\n",msg_num);
        break;
    default:
        fprintf(outfile,"%s",buffer);
        break;
)
else
(
    if (line == 1)
    (
        t = time(NULL);
        fprintf(outfile,"Date: %s",ctime(&t));
        fprintf(outfile,"From: tfmobox\n");
        fprintf(outfile,"Subject: Telefind Network Message\n");
        fprintf(outfile,"To: <Name here>\n");
        if (msg_num_opt)
        (
            fprintf(outfile,"Message #%s\n",msg_num);
            fprintf(outfile,"%s",buffer+3);
        )
        else
            fprintf(outfile,"%s",buffer);
    )
    else
        fprintf(outfile,"%s",buffer);
)

if (strcmp(buffer,DELIMITER) == 0)
(
    msg_num_opt = line = attmail = 0;
)

line ++;
)

```

```

Author:      MICHAEL P. SCHKE, SR.
              03/13/91

Program:     SAFARI3.C
Purpose:     TO EXTRACT MESSAGES FROM A TELEFIND PAGER
              VIA IN RS-232 PORT ON A PC

Compiler:    TURBO C++ 1.0
Memory Model: SMALL

```

```

*/

#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include "safari.h"

/*      CONSTANTS      */

#define DTR_HI      0x01
#define DTR_LO      0xfe
#define RTS_HI      0x02
#define RTS_LO      0xfd
#define DSR_HI      0x20
#define RING_IN      0x40
#define CD_HI        0x80
#define FIVE_TICK    5
#define FIVE_SEC     96
#define TWELVE_SEC   220
#define LOG_FILE     "LOG"
#define INTRO_STRING "Please standby, retrieving messages ..."

/*      FUNCTION PROTOTYPES      */

int beep(void);
void busyoff(void);
void busyon(void);
void disoff(void);
void dison(void);
int link(void);
void print_message(void);
int rxdata(void);
int strobe(void);
int strobe_data(void);
unsigned ticks(void);
int timeout(unsigned start, int delay);

/*      VARIABLE DECLARATIONS      */

char pager_buffer[511];
int com_base, control_reg, status_reg, log_flag;
FILE *log_file;

void main(int num_arg, char **args)
(
    unsigned start;
    int restart,x;

    com_base = 0x3f8;      /* use com 1 unless command line denotes otherwise */

    /*      get command line arguments      */

```

all command line arguments begin with a single '-' and must be separated by a single space between each other and the program name

```
-1      Use COM port 1
-2      Use COM port 2
-F      Log all activity to a file named LOG      */
```

```
if (num_arg > 1)
{
    for (x=1; x<num_arg; x++)
    {
        if (strcmp(args[x],"-1") == 0)
            com_base = 0x3f8;
        if (strcmp(args[x],"-2") == 0)
            com_base = 0x2f8;
        if (strcmp(args[x],"-F") == 0)
            log_flag = 1;
    }
}

if (log_flag)
    if ((log_file = fopen(LOG_FILE,"at")) == NULL)
        printf("Unable to open LOG\n");
```

```
control_reg = com_base + 4;
status_reg = com_base + 6;
```

```
clrscr();
```

```
if ((link() == 0)      /* is pager attached ?      */
{
    printf("Please attach Message Receiver \n");
    exit(0);
}
```

```
busyon();      /* start busy at logic high      */
```

```
if (log_flag)
    fprintf(log_file,"Initiating process \n");
```

```
printf("%s\n",INTRO_STRING);
```

```
dison();      /* push display button */
```

```
sleep(2);
```

```
do
```

```
{
```

```
    start = ticks();
```

```
    restart = 0;
```

```
    do
```

```
    {
```

```
        if (beep())
```

```
        {
```

```
            print_message();
```

```
            restart = 1;
```

```
            start -= TWELVE_SEC;
```

```
            break;
```

```
        }
```

```
    }
```

```
    /* hold display button for 12 seconds */
```

```
    while(! timeout(start,TWELVE_SEC));
```

```
}
```

```
while(restart);
```

```
disoff();      /* release the display button */
```

```
if (log_flag)
```

```
{
```

```
    fprintf(log_file,"Process Complete \n");
```

```

        fclose(log_file);
    }

}

/*          pager beep          */
int beep(void)
{

/*      accesses the RI line via the Status Register
       which is activated when the pager beeps      */

    unsigned start;

    start = ticks();
    while ( ! timeout(start,FIVE_TICK))
    {
        if ((inportb(status_reg) & RING_IN) == 0 )
            return(1);
    }
    return(0);
}

/*      busyon & busyoff toggle the DTR line via the
       Control Register to strobe in data from the pager      */
void busyoff(void)
{
    outportb(control_reg,inportb(control_reg) | DTR_HI);
}

void busyon(void)
{
    outportb(control_reg,inportb(control_reg) & DTR_LO);
}

/*      dison & disoff toggle the RTS line via the Control Register
       to simulate the pressing of the display button on the pager      */
void dison(void)
{
    outportb(control_reg,inportb(control_reg) | RTS_HI);
}

void disoff(void)
{
    outportb(control_reg,inportb(control_reg) & RTS_LO);
}

int link(void)
{

/*      accesses the CD line via the Status Register
       which is logic high when pager is connected      */

    if ((inportb(status_reg) & CD_HI) == 0)
        return(0);
    return(1);
}

void print_message(void)
{
    FILE *file;
    unsigned start;
    int x,y=0,z=0,chr,bit;

```

```

busyoff(); /* ready to accept pager data */

/* read until end code received */
while (chr != 3)
(
    chr = 0;
    start = ticks();

    /* wait for start bit */

    do
    (
        bit = strobe();
        if (bit == 0)
            break;
    )
    while (!timeout(start,FIVE_SEC));

    if (bit)
    (
        if (log_flag)
            fprintf(log_file,"Transmission Error, recheck connection\n");
        disoff();
        exit(0);
    )

    /* strobe out 8 bit data */
    for (x=1; x<9; x++)
    (
        chr <<= 1;
        chr += bit = strobe_data();
    )

    /* clear out stop bits */
    for (x=1;x<3;x++)
    (
        strobe_data();
    )

    /* extract start and end codes from message

    pager signon      02, 18, 00, 33
    pager signoff     03 */

    if ((y > 3) && (chr != 3))
    (
        /* pager characters 96 and 97 are converted to
        0xFA and 0xFB to display on pager */

        if (chr == 0xfa) /* convert to CR */
            chr = '\n';
        if (chr == 0xfb) /* convert to TAB */
            chr = 0x09;

        pager_buffer[z] = chr;
        z ++;
    )
    y ++;
)

pager_buffer[z] = '\0'; /* null terminate */

busyon(); /* finished receiving data */

```



```

if (log_flag)
    fprintf(log_file,"%s\n",pager_buffer);

if ((file = fopen(ATT_EMAIL_FILE, "at")) == NULL)
    fprintf(log_file,"Unable to open IFMOBOX.THP\n");
else
{
    fprintf(file,"%s\n",pager_buffer);
    fprintf(file,"%s",DELIMITER);
    fclose(file);
}

start = ticks();
while(!timeout(start,FIVE_SEC))
{
    /* wait for erase beep */
    if (beep()) break;
}
sleep(1); /* wait one more second */
}

```

```

int rxdata(void)
{

```

```

/* accesses the DSR line via the Status Register
   which returns the bits value */
if (inportb(status_reg) & DSR_HI)
    return(0);
return(1);
}

```

```

int strobe(void)
{

```

```

    int bit;

    busyon();
    delay(1);
    busyoff();
    delay(4);
    bit = rxdata();
    return(bit);
}

```

```

int strobe_data(void)
{

```

```

    int bit;

    busyon();
    delay(2);
    bit = rxdata();
    busyoff();
    delay(1);
    return(bit);
}

```

```

unsigned ticks(void)
{

```

```

    /* returns timer ticks (approx. 18.2/sec)
       using only lower registers */

    union REGS in,out;

    in.x.ax = 0x0;
    int86(0x1a,&in,&out);
    return(out.x.dx);
}

```

```
timeout(unsigned start, int delay)
```

```
/*      used for timing events of up to approx. 1 hour.
      used in conjunction w/ticks()          */
```

unsigned current;

```
current = ticks();
if (start <= current && (start + delay) < current)
    return(1);
if (start > current && (start - 65535 + delay) < current)
    return(1);
return(0);
```

00445409 = 1206999

0945403.12069  
0945403.12069

```

/* mark the end of the command line you built,so you can add ending
   delimiter */
sys_command[i] = NULL;
/* add the ending quote for the users message so shell wont
   interepert special characters */
strcat(sys_command, "\\");
/* execute command you built */
system(sys_command);

printf("sending message: %s\n", sys_command);

}
else {
    if(strlen(msg) == 0 ) {
        return(0);
    }
    /* print error for invalid message length */
    printf("telemail error: invalid message length: %s\n", msg);
    return(0);
}

return(i);
}

.....
*   function: getline(hold-buffer, input-file-pointer)
*   arguments: pointer to buffer where line read will be heald,
*               file pointer to input file
*   description: reads 1 line of text from the input line and stores the
*               line read into the buffer passed.
*   returns: -1 if EOF or number of characters read in
*
...../
getline(buff, fp)
char *buff;
FILE *fp;
{
    int ch, cnt;

    /* keep on reading characetrns from file so long as end of file not
       reached or char is the end of line */
    for(cnt = 0; ((ch = fgetc(fp)) != EOF) && ch != '\n'; cnt++) {
        /* MOD BY OT 11/29/90 convert tab to space */
        /* convert tabs to single space */
        if(ch == 9) {
            ch = ' ';
        }
        /* MOD BY OT 11/29/90 dont allow control char */
        /* only load in ascii characters */
        if(isprint(ch) != 0) {
            buff[cnt] = ch;
        }
        else {
            /* turn control characters to spaces */
            buff[cnt] = ' ';
        }
    }

    /* mark the end of the buffer you built */
    buff[cnt] = '\0';
}

```

66902T 60453460  
66902T 60453460

```

/*****
*
*   function: send_mesg(message-pointer)
*   arguments: pointer to text message(capcode,text) to be sent
*   description: takes passed message text makes sure the first 8 positions
*                 are numeric(capcode). it builds and executes the network
*                 send command(netsend.sh) to sedn the message passed.
*   returns: 0 if not sent otherwise the number of characters sent out
*
*****/
int send_mesg(mesg)
char *mesg;
{
    char sys_command[700];
    int i;
    int ch;
    char *mesg_ptr;

    /* left justify the message passed to remove leading spaces */
    strljust(mesg, 512);
    /* trim off trailing blank spaces from the message */
    strtrim(mesg);

    /* make sure you have a capcode at least */
    if(strlen(mesg) > 8) {

        /* start to build the command to be executed to send message retrieved
           from the mail box */
        strcpy(sys_command, "netsend.sh ");

        /* loop while still more characters in the message */
        for(mesg_ptr = mesg, i = 11; *mesg_ptr != NULL; i++, mesg_ptr++) {

            /* make sure the first 8 positions of the message are numeric */
            if((i < 19) && (*mesg_ptr < '0' || *mesg_ptr > '9')) {
                printf("telemail error: invalid capcode: %s\n", mesg);
                return 0;
            }

            /* is the user didnt separte capcode & message then insert a
               space into the command */
            if(i == 19 && *mesg_ptr != ' ') {
                sys_command[19] = ' ';
                i = 20;
            }

            /* enclose the users message with ' so shell wont interpet
               special characters */
            if(i == 20) {
                sys_command[20] = '\'';
                i = 21;
            }

            /* put the character from the message onto to the
               command to be executed */
            sys_command[i] = *mesg_ptr;
        }
    }
}

```

0050027 00454460  
0050027 00454460

```
/* since your just starting clear the message area */
memset(msg, NULL, MAXMSGLEN);

/* keep on getting lines from the file until you reach end of file */
while(getline(buff, fp) != -1) {

    /* every mail message start with the word "From " */
    if(strncmp(buff, "From ", 5) == 0) {
        /* set flag telling you are currently going thru mail header
           so you dont add it to the message */
        in_header = 1;
        /* call routine to the last message if any exists */
        send_mesg(msg);
        continue;
    }

    /* a mail header end with the following string */
    if(strncmp(buff, "Content-Length:", 15) == 0) {
        /* turn off flag so you know you are no longer in mail
           message header */
        in_header = 0;
        /* clear the old message since this is a new one */
        memset(msg, NULL, MAXMSGLEN);
        continue;
    }

    /* if the line you are now reading in not part of the mail header
       add it to the message */
    if(in_header == 0) {
        strljust(buff, 512);
        strtrim(buff);
        /* make sure you dont add more than the message length */
        if( (strlen(buff) + strlen(msg)) < MAXMSGLEN) {
            strcat(msg, " ");
            strcat(msg, buff);
        }
    }
}

/* end of read line while */

/* send the last message in the file */
send_mesg(msg);
}
```